L6: Entry 1 of 1 File: USPT Aug 13, 2002

DOCUMENT-IDENTIFIER: US 6434681 B1
TITLE: Snapshot copy facility for a data storage system permitting continued host read/write access

Abstract Text (1):
A snapshot copy of a production data set is maintained while a host may continue write access to the production data set. The data storage system responds to a host request to write to a storage location of the production data set by checking whether or not the storage-location has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set. In the preferred implementation, the data storage system allocates to the snapshot copy a bit map to indicate storage locations in the production data set that have been modified, and a list of pointers to allocated storage locations for the snapshot copy. The snapshot copy facility is useful so that a host write operation upon a storage location being backed up need not be delayed until original data in the storage location is written to secondary storage. The snapshot copy facility is also useful for other applications such as transaction processing and debugging.

Brief Summary Text (2):
The present invention relates generally to computer data storage, and more particularly, to a snapshot copy facility for a data storage system that permits continued host read/write access to data storage that has been snapshot copied.

Brief Summary Text (4):
Snapshot copies of a data set such as a file or storage volume have been used for a variety of data processing and storage management functions such as storage backup, transaction processing, and software debugging.

Brief Summary Text (5):
A known way of making a snapshot copy is to respond to a snapshot copy request by invoking a task that copies data from a production data set to a snapshot copy data set. A host processor, however, cannot write new data to a storage location in the production data set until the original contents of the storage location have been copied to the snapshot copy data set.

Brief Summary Text (6):
Another way of making a snapshot copy of a data set is to allocate storage to modified versions of physical storage units, and to retain the original versions of the physical storage units as a snapshot copy. Whenever the host writes new data to a storage location in a production data set, the original data is read from the storage location containing the most current version, modified, and written to a different storage location. This is known in the art as a "log structured file" approach. See, for example, Douglis et al. "Log Structured File Systems," COMPCON

89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, p. 124-129, incorporated herein by reference, and Rosenblum et al., "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 1, February 1992, p. 26-52, incorporated herein by reference.

Brief Summary Text (11):
In practice, it has been found that the technique of storing data in logical storage volumes having relatively permanent physical storage locations in a data storage system has a competitive advantage over the more complex log structured file approach. For providing storage backup, however, the relatively permanent physical storage locations of the logical storage volumes has introduced a significant delay when host write access to storage locations containing original data is delayed until the original data are transmitted to a backup storage device. It has been found that this delay can be reduced by providing a snapshot copy facility in the data storage system. This snapshot copy facility is useful for other applications, such as transaction processing and debugging.

Brief Summary Text (12):
In accordance with a first aspect of the invention, there is provided a method of maintaining in data storage of a data storage system a snapshot copy of a production data set including a multiplicity of storage locations in the data storage. The production data set is accessible to a host processor for read/write access during maintenance of the snapshot copy. The snapshot copy includes data existing in the production data set at a time when the snapshot copy is created. The method includes the data storage system responding to a request from the host processor for a write operation upon a storage location of the production data set. The data storage system responds by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set.

Brief Summary Text (13):
In accordance with another aspect, the invention provides a method of maintaining in data storage of a data storage system a snapshot copy of a production data set including a multiplicity of storage locations in the data storage. The production data set is accessible to a host processor for read/write access during maintenance of the snapshot copy. The snapshot copy includes data existing in the production data set at a time when the snapshot copy is created. The method includes the data storage system allocating to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not each storage location has been modified since the time when the snapshot copy is created. The method further includes the data storage system responding to a request from the host processor for a write operation upon a storage location of the production data set. The data storage system responds by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location

of the snapshot copy, performing the write operation upon the storage location of the production data set.

Brief Summary Text (14):
In accordance with yet another aspect, the invention provides a data storage system including data storage and at least one data processor responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage. The data processor is programmed to maintain in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created. Moreover, the data processor is programmed to respond to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set.

Brief Summary Text (15):
In accordance with still another aspect, the invention provides a data storage system including data storage and at least one data processor responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage. The data processor is programmed to maintain in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created. The data processor is also programmed to allocate to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not each storage location has been modified since the time when the snapshot copy is created. The data processor is further programmed to respond to a request from the host processor for a write operation upon a storage location of the production data set by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set.

Brief Summary Text (16):
In accordance with a further aspect, the invention provides a program storage device containing a program for a data processor in a data storage system including data storage. The data processor is responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage. The program is executable by the data processor for maintaining in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created. The program is also executable by the data processor for responding to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the

snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set.

Brief Summary Text (17):
In accordance with a final aspect, the invention provides a program storage device containing a program for a data processor in a data storage system including data storage. The data processor is responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage. The program is executable by the data processor for maintaining in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created. The program is also executable by the data processor for allocating to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not each storage location has been modified since the time when the snapshot copy is created. Moreover, the program is executable by the data processor for responding to a request from the host processor for a write operation upon a storage location of the production data set by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set.

Drawing Description Text (7):
FIG. 5 is a block diagram of data structures which are included in a snapshot copy facility of the primary data storage subsystem of FIG. 1;

Drawing Description Text (8):
FIG. 6 is a schematic diagram of a preferred construction for a list of pointers to tracks in snapshot disks for the snapshot copy facility as shown in FIG. 5;

Drawing Description Text (9):
FIG. 7A is a flow chart of programming of the port adapters in the primary data storage subsystem of FIG. 3 for creating a snapshot of a production volume extent;

Drawing Description Text (10):
FIG. 7B is a flow chart of programming of the port adapters in the primary data storage subsystem of FIG. 3 for access to a production volume that is in a snapshot mode;

Drawing Description Text (11):
FIG. 8A is a flow chart of programming of the remote link adapters in the primary data storage subsystem of FIG. 3 for a preferred implementation of the snapshot copy facility as introduced in FIG. 5;

Drawing Description Text (13):
FIG. 8C is a flow chart of programming of the remote link adapters in the primary

data storage subsystem of FIG. 3 for an alternative implementation of the snapshot copy facility as introduced in FIG. 5;

Drawing Description Text (22):
FIG. 17 is a flow chart of port adapter programming to permit access to a specified track in a snapshot copy of a production volume.

Detailed Description Text (7):
In response to a backup command from the host 20, the primary data storage subsystem 21 accesses a primary directory 26 to find data of the physical storage unit specified by the backup command in order to initiate a process of copying the data from the primary storage 27 to the secondary storage 29 of the secondary data storage subsystem 22. Preferably, the primary directory 26 is constructed in such a way that the host can continue to access the primary storage 27 concurrently with the copying process. For example, in response to the backup command from the host 20, the primary data storage subsystem creates an "instant snapshot copy" of the specified physical storage unit, and this instant snapshot copy is protected from modification by the host 20 while the instant snapshot copy is being written to the secondary storage 29. There are a number of ways that such an instant snapshot copy can be created, depending on the way that the primary directory is organized.

Detailed Description Text (8):
One way of organizing the primary directory 26 is to associate a set of flags and mapping information with each physical storage unit, for example as described in Yanai et al., U.S. Pat. No. 5,206,939, issued Apr. 27, 1993, and incorporated herein by reference. In order to create an instant snapshot copy, a remote copy pending flag is associated with each physical storage unit. When the primary data storage subsystem 21 receives a backup command from the host 20, the primary data storage subsystem sets the remote copy pending flag, and thereafter the host can concurrently access the primary storage 27 while data is being copied from the physical storage unit to the secondary storage 29. However, before the primary data storage subsystem accesses the primary storage 27 to modify any data in response to a request from the host 20, the primary data storage subsystem first inspects the remote copy pending flag of the physical storage unit to be modified, and if the remote copy pending flag is set, the primary data storage subsystem must copy the data of the physical storage unit from the primary storage 27 to the secondary storage 29 and reset the remote copy flag, before modifying the data in the primary data storage subsystem. Unless there is such a request from the host for modification of data marked as "remote copy pending," the copying of data from the primary storage 27 to the secondary storage 29 is performed as a background process relative to host requests.

Detailed Description Text (9):
Another way of organizing the primary directory 26 is to maintain lists of pointers to primary storage locations of old and new versions of physical storage units. Whenever data in the primary storage is modified, the data is read from the primary storage locations containing the most current version, modified, and written to a different set of primary storage locations. This is known in the art as a "log structured file" approach. See, for example, Douglis et al. "Log Structured File Systems," COMPCON 89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, p. 124-129, incorporated herein by reference, and Rosenblum et al., "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 1, February 1992, p. 26-52, incorporated herein by reference. In order to create an instant snapshot copy, the primary directory 26 includes a respective remote copy pending flag associated with the pointer to each version of each physical storage unit. In response to a backup command from the host 20, the primary data storage subsystem sets the remote copy flag. Thereafter, the primary data storage subsystem can modify the data of the physical storage unit in the primary storage in the usual fashion concurrently with the copying of a snapshotted version of the data to the secondary storage, because the new version and the

snapshotted version of the physical storage unit are stored in a different set of primary storage locations. Instead of being used to temporarily delay any modification access to a physical storage unit, the remote copy flag is used to indicate whether or not the set of primary storage locations associated with an old version of a physical storage unit can be de-allocated after a modification access.

Detailed Description Text (11):
In the preferred implementation of the data processing system of FIG. 1, the primary data storage subsystem 21 includes a snapshot copy facility 69. The snapshot copy facility 69 includes a stored program that is executed by data processors in the primary data storage subsystem as described below with reference to FIGS. 5 to 8. This stored program is a component of what is known as microcode for the primary data storage subsystem. The microcode can be down-loaded into program memory of the primary data storage subsystem from a conventional program storage device such as a floppy disk.

Detailed Description Text (12):
Regardless of how the primary directory 26 is organized and how the instant snapshot process is performed, it is possible for the secondary storage 29 to contain more than one version of backup data for the same physical storage unit. In order to distinguish between different versions of backup data for the same physical storage unit, the primary data storage subsystem 21 appends an identification tag to the backup data transmitted from the primary data storage subsystem to the secondary data storage subsystem 22. The tag, for example, is supplied by the host 20 in the backup command transmitted by the host to the primary data storage subsystem 21. The tag could also include a date-time stamp generated by the primary data storage subsystem. In the secondary data storage subsystem 22, the tag associated with each version of backup data is stored in a secondary directory 28, which further includes a record of a set of locations of the secondary storage 29 in which the version of backup data is stored.

Detailed Description Text (16):
With reference to FIG. 2, a data network 30 such as a Fibre Channel loop links a multiplicity of hosts 31, 32, 33 to a number of primary data storage subsystems 41, 42. The hosts 31, 32, 33, for example, are workstations of respective users 34, 35, 36. The user 35 is a system manager responsible for configuring the data storage subsystems 41, 42 and ensuring that the data storage and backup needs of the users are satisfied. Each of the hosts has a copy of backup software 37, 38, 39 similar to the backup software 24 described above with reference to FIG. 1. The primary data storage subsystems 41, 42 each have a respective primary directory 44, 46, respective primary storage 45, 47, and respective snapshot copy facilities 67, 68. The primary data storage subsystems 41, 42 are each similar to the primary data storage subsystem 21 of FIG. 1. The primary data storage subsystems 41, 42 share a secondary data storage subsystem 43. The secondary data storage subsystem 43 has a secondary directory 48, secondary storage 49, and a disk caching facility 69. The secondary data storage subsystem 43 is similar to the secondary data storage subsystem 22 of FIG. 1, but it further includes independent, dedicated data links 93 and 94 to each of the primary data storage subsystems 41 and 42, and a data link 92 to the data network 30. The dedicated links 93, 94 are used for transferring backup data between the respective primary data storage subsystems 41, 42 and the secondary data storage subsystem 43. The data link 92 permits the system manager 35 to access the secondary data storage subsystem 43 for data storage management and diagnostic purposes.

Detailed Description Text (31):
II. Snapshot Copy Facility.

Detailed Description Text (32):
As described above with reference to FIGS. 1 to 4, a primary data storage subsystem

and a secondary data storage subsystem have been constructed to rapidly respond to a backup request. The primary data storage subsystem responds by performing a snapshot copy, and transferring backup data from the snapshot copy to the secondary storage subsystem.

Detailed Description Text (33):
With reference to FIG. 5, there are shown a number of data structures that are located in the primary data storage subsystem and are used by the snapshot copy facility (69 in FIG. 1) of the primary data storage subsystem. In the example of FIG. 5, snapshot copies have been made of two production volumes 101 and 102. The snapshot copies are shown as they would exist some time after the primary storage subsystem has received a first backup command for backing up an "extent" of the production volume 101, and some time after the primary data storage subsystem has received a second backup command for backing up an extent of the production volume 102. An "extent" of a production volume is a set of contiguous tracks of the production volume, as specified, for example, by a beginning track number and an ending track number. Since receipt of the first backup command, a host has modified tracks A and B of the production volume 101, and since receipt of the second backup command, a host has modified tracks G and H of the production volume 102. Before the first modification of each track in the snapshotted production volume 101 or 102, however, the primary data storage subsystem copies the original contents of the production volume (i.e., the contents existing at the time of the snapshot) to a track in a snapshot volume 103, 104. For example, before the primary data storage subsystem modifies track A of the production volume 101, the original contents of track A are copied to track 0 of the snapshot volume 103. In a similar fashion, track 1 of the snapshot volume 103 contains the original contents of track B of the production volume 101, track 2 of the snapshot volume 103 contains the original contents of track G of the production volume 102, and track 3 of the snapshot volume 103 contains the original contents of track H of the production volume 102.

Detailed Description Text (35):
The data structures in FIG. 5 also include, for each snapshotted production volume extent, a list of pointers 106, 108 to tracks in the snapshot volumes that contain original data of the snapshot. The data structures also include, for each snapshot volume, a list of pointers 109, 110 to free tracks. When a track is copied from an extent of a production volume to a snapshot volume, a pointer to the track is taken from the list of pointers to free tracks in the snapshot volume and added to the list of pointers for the extent.

Detailed Description Text (36):
When a snapshot copy of a production volume is first created, the directory to the production volume is locked to host access while the bit map and list of pointers to snapshot tracks are allocated and initialized for the snapshot copy, and then the directory is unlocked.

Detailed Description Text (37):
The data structures for the snapshot copy facility as shown in FIG. 5 permit the production volumes to be configured and indexed in the usual fashion. The tracks of the snapshot copy for an extent can be obtained by scanning the bit map for the extent and accessing the production volume for a track having its respective bit not set in the bit map, and otherwise accessing the track in the snapshot volume using a pointer from the list of pointers for the extent if the respective bit for the track is set. The data structures used by the snapshot copy facility include information associating the tracks for the extent with their respective pointers in the list.

Detailed Description Text (38):
Although not necessary for making backup copies, the data structures associated with the snapshot copy facility may include an index to the snapshot tracks for each snapshot copy of a production volume. Such an index, for example, would be

desirable if the snapshot copy facility were used for providing specified snapshot data to a distributed transaction processing system. It could also be useful if it were desired to transmit snapshot copy data from the primary storage subsystem in sequential track order, or to de-allocate specified snapshot tracks. As shown in FIG. 5, for example, an index 111 is provided to locate, in the list of pointers 106, the pointer to any snapshot track containing original data from a specified track in the production volume 101. Also, an index 112 is provided to locate, in the list of pointers 108, the pointer to any snapshot track containing original data from a specified track in the production volume 102. In other words, each index 111, 112 functions as a kind of translation table, to translate a production volume track number to a corresponding snapshot volume number and snapshot volume track number.

Detailed Description Text (39):
Although the indices 111 and 112 can be constructed in various ways, in many applications only a small fraction of the production volume tracks will have corresponding snapshot volume tracks. In this situation, a conventional hash table index is an efficient implementation for the indices 111, 112. For the index 111, for example, such a conventional hash table index includes, for each possible hashing of the production volume track number, a list of pointers to corresponding entries in the list 106. Therefore, given a specified production volume track number, any corresponding snapshot track is found by hashing the production volume track number (i.e., obtaining the remainder of the division of the production volume track number by a predetermined number of possible hashings), and then searching the corresponding entries in the list 106 to locate an entry for the specified production volume track number.

Detailed Description Text (40):
As shown in FIG. 6, the list of pointers 106 includes heading information that specifies the list's production volume extent. This heading information includes a logical device number (DEVICE_NUM), a first track number for the extent (FIRST_TRACK), and a last track number for the extent (LAST.sub.13 TRACK). Each entry in the list includes a snapshot volume number (X), a production track number (A, B), and a snapshot track number (0, 1). In this example, the list is maintained as a singly-linked list, so that the header information includes a pointer to a first entry in the list (or has a value of zero if the list is empty), and each entry in the list includes a pointer to a next entry in the list (and this pointer has a value of zero for the last entry in the list).

Detailed Description Text (41):
When the snapshot copy facility is used to facilitate backup operations, it is desirable to use a singly-linked list of pointers 106 instead of simply a list in the form of an array in order to conserve memory for the list because the length of the list is highly variable and it is possible that multiple snapshot copies may be in existence simultaneously. For example, the tape library unit has multiple read/write stations (81, 82, 83, 84 in FIG. 4) which may be writing backup data simultaneously to different respective tape cassettes, and each production volume extent is written to a respective tape cassette.

Detailed Description Text (42):
Referring to FIG. 7A, there is shown a flow chart of a software procedure programmed in the port adapters of a primary data storage subsystem for creating a snapshot copy of a production volume. The procedure of FIG. 7A is invoked, for example, when the port adapter receives a command from a host requesting backup of a specified production volume extent. In the first step 120 of FIG. 7A, the port adapter allocates the snapshot data structures, including a bit map and a list of pointers to snapshot tracks for the extent. An index to the pointers may also be allocated. Then in step 121, the port adapter inserts, into the volume directory entry for the production volume, a pointer to the snapshot data structures. This pointer, for example, is zero in the volume directory entry for any volume which

does not have a snapshot copy. The volume directory entry could also have a field specifying a minimum track number for the production volume extent to be snapshotted, and a maximum track number for the production volume extent to be snapshotted. After step 121, the snapshot creation procedure is finished.

Detailed Description Text (43):
Referring to FIG. 7B, there is shown a flowchart of a software procedure programmed into the port adapters for accessing a production volume in a snapshot mode. This procedure is invoked whenever a host processor requests a write operation to a production volume for which a snapshot could have been created. In the first step 122 the port adapter checks the volume director entry to determine whether a snapshot currently exists for the production volume, and if so whether the write operation is upon a track within the production volume extent of the snapshot. If the access to the production volume is not a write to a track within the production volume extent of the snapshot, then execution branches to step 123 to access the track in the production volume, and then the procedure of FIG. 7B is finished. Otherwise, if the access to the production volume is a write to a track within the production volume extent of the snapshot, then execution continues from step 122 to step 124. In step 124, the port adapter inspects the bit for the track in the bit map for the extent. If the bit is set, then execution branches to step 123 to access the track in the production volume extent, and then the procedure of FIG. 7B is finished. Otherwise, if the bit for the track is not set in the bit map, then execution continues from step 124 to step 125. In step 125 the port adapter obtains a pointer to a free track in one of the snapshot volumes. For example, the port adapter first checks whether the list of pointers 109 for a first volume 103 is empty, and if so, it then checks the list of pointers for other snapshot volumes such as the volume 104 in FIG. 5.

Detailed Description Text (44):
Once the port adapter finds a non-empty list of pointers to free tracks of a snapshot volume, it removes a pointer from the list. In step 126 the port adapter copies the track to be modified from the production volume to the snapshot volume track specified by the pointer that was taken from the list of pointers to free tracks in the snapshot volume. In step 127 the port adapter inserts the pointer into the list of snapshot track pointers for the extent, and also inserts into this list entry an identifier for the snapshot volume and an identifier for the track in the production volume extent. Then in step 128 the port adapter sets the bit in the bit map to indicate that the track is being modified. Finally, in step 129 the port adapter writes new data to the track in the production volume. After step 129, the procedure of FIG. 7B is finished.

Detailed Description Text (45):
With reference to FIG. 8A, there is shown a flow chart of a software procedure programmed in a remote link adapter of a primary storage subsystem for implementation of the snapshot copy facility. This procedure is invoked in response to a message from a port adapter when the port adapter receives a backup command from a host. In a first step 131, the remote link adapter sets a track copy pointer to point to the first track in the production volume extent. Then in step 132 the remote link adapter checks whether the bit for the track is set in the bit map for the production volume extent. If not, then in step 133 the remote link adapter copies the track from the production volume to secondary storage. Execution continues from step 133 to step 134. Execution also branches from step 132 to step 134 when the remote link adapter finds in step 132 that the bit for the track has been set in the bit map. In step 134, the remote link adapter checks whether the track copy pointer is pointing to the end of the production volume extent. If not, execution continues from step 134 to step 135. In step 135 the track copy pointer is incremented to point to the next track in the extent, and execution loops back to step 132.

Detailed Description Text (46):

If in step 134 the remote link adapter finds that the track copy pointer is pointing to the end of the production volume extent, then execution branches to step 136. In step 136, the remote link adapter deallocates the bit map. Then in step 137, for each entry in the list of pointers to snapshot tracks, the remote link adapter copies the track from the snapshot volume to secondary storage, and removes the entry from the list. Finally, in step 138, the remote link adapter deallocates the list of pointers to snapshot tracks for the extent, and the backup operation is finished.

Detailed Description Text (47):
The remote link adapter routine of FIG. 8A may copy the tracks of snapshot data in either a synchronous fashion or an asynchronous fashion from the primary data storage subsystem to the secondary storage subsystem. If the copying is done in a synchronous fashion, the procedure in FIG. 8A would not proceed from step 133 to step 134 until the remote link adapter would receive confirmation from the secondary storage that the track has in fact been written to the secondary storage. Preferably, however, the copying is done in an asynchronous fashion, in which the track is transmitted from the production volume to secondary storage in step 133 and execution then continues to step 134 without waiting for confirmation that the that the track has been written to the secondary storage. Once the remote link adapter receives confirmation that a track has been written from the production volume to secondary storage, the bit for the track in the bit map can be set to avoid any delay if and when the host makes any write access to the production volume track before the entire production volume extent has been copied to secondary storage. Also, if a backup copy is made in an asynchronous fashion, the bit map is not deallocated in step 136 until after receipt of confirmation that all of the tracks sent from the production volume to secondary storage in step 133 have actually been written to secondary storage.

Detailed Description Text (48):
To facilitate backup copying to secondary storage in an asynchronous fashion, the remote link adapter can be programmed as shown in FIG. 8B to respond to a command from the secondary data storage system indicating that a range of specified tracks have been in fact copied to secondary storage and therefore the specified tracks can be deallocated from the snapshot copy. The range of specified tracks, for example, is a small subset of the tracks in the production volume extent being backed up. In this case, the command from the secondary storage is a request to "partially free" the snapshot copy. The deallocation of the specified range of tracks will prevent copying of production volume tracks in the specified range to snapshot tracks if a host subsequently writes to the tracks for the first time after the creation of the snapshot copy, and will free any snapshot volume tracks that have been allocated to production volume tracks within the specified range.

Detailed Description Text (49):
In a first step 221 of FIG. 8B, the remote link adapter sets a deallocation track pointer to the first track in the specified range of tracks. Then in step 222, execution branches depending on the state of the bit for the track in the bit map. If the bit for the track is set in the bit map, then execution branches from step 222 to step 223. In step 223, any snapshot volume track allocated for the production volume track is deallocated by removing the snapshot track pointer from the list for the snapshot volume and returning the snapshot track pointer to the free list for the snapshot volume. If in step 222 the bit for the track is not set in the bit map, then execution continues from step 222 to step 224. In step 224, the bit for the track is set in the bit map. Therefore, if a host subsequently writes to the track, a copy of the original contents of the track will not be copied to a snapshot volume track. After step 223 or 224, execution continues to step 225. In step 225, the deallocation track pointer is compared to the end of the specified range. If the deallocation track pointer is at the end of the specified range, then the procedure is finished. If not, execution continues to step 226. In step 226, the deallocation track pointer is incremented, and execution loops back

to step 222.

Detailed Description Text (51):
Alternatively, the tracks can be copied from a snapshot to secondary storage in a sequential fashion. In an example of such an alternative procedure, as shown in the flow chart of FIG. 8C, the link adapter is programmed to respond to a backup request by sequentially incrementing a track copy pointer from the beginning track in a production volume extent to an ending track in the production volume extent. In a first step 241, the ink adapter sets the track copy pointer to point to the first track in the production volume extent. Then, in step 242, for the track pointed to by the track copy pointer, the link adapter indexes the bit map for the snapshot. If the bit for the track is set in the bitmap for the snapshot, then execution continues from step 242 to step 243. In step 243, the link adapter accesses the index to the snapshot tracks to translate the track number specified by the track copy pointer to a snapshot volume number and a snapshot track number. Then in step 244, the link adapter reads the snapshot track from the snapshot volume and transmits this track of backup data from the snapshot track to the secondary storage subsystem.

Detailed Description Text (52):
If the bit for the track is not set in the bit map for the snapshot, then execution branches from step 242 to step 245. In step 245, the link adapter reads the track specified by the track copy pointer from the production volume and transmits this track of backup data to the secondary storage subsystem. After step 244 or 245, execution continues to step 246. In step 246, if the track copy pointer has not reached the end of the production volume extent, then execution branches to step 247. In step 247, the track copy pointer is incremented, and execution loops back to step 242. Once the track copy pointer reaches the end of the production volume extent in step 246, the procedure of FIG. 8C is finished.

Detailed Description Text (53):
With reference to FIG. 9, there is shown a format of a data record on the backup tape. In accordance with a conventional tape record, the record shown in FIG. 9 includes, after an inter-record gap 141, a synchronization code 142, a record number 143, record data (fields 144, 145, and 146), and finally an error correction code 147 preceding another inter-record gap 148. Since the records on the backup tape are not necessarily sequential by track number, it is desirable for each record to include a track number 145 in the record data. Also, it is assumed that each tape cassette will include data from only one production volume extent. It is possible, however, that a tape cassette could be used, at various times, to store more than one version of data from the same production volume extent. Therefore, it is also desirable for the data of the record on tape to include a version identifier 144, such as a date/time stamp when the snapshot copy was made. As shown in FIG. 9, the data portion of the record on the backup tape includes the version identifier 144 followed by the track number 145 and track data 146.

Detailed Description Text (65):
IV. Additional Applications of Snapshot Copy Facility

Detailed Description Text (66):
As described above, the snapshot copy facility is useful in a primary storage system so that a host write operation upon a storage location of a volume extent being backed up need not be delayed until original data in the storage location is written to secondary storage. The snapshot copy facility, however, can be used for other applications such as transaction processing and debugging.

Detailed Description Text (68):
The snapshot copy facility as described above can simplify considerably the implementation, host processing time, and storage requirements for the "commit" and "abort" operations. For example, the "commit" operation simply flushes any file

modifications to storage, and then creates or clears a snapshot copy for each of the files. The "abort" operation checks whether the snapshots for the files have all been created or cleared in sequence, for example, by inspecting time stamps associated with each snapshot. If the snapshots for the files have not all been created or cleared in sequence, then a failure occurred during the process of creating or clearing the snapshot copies, and the recovery operation simply finishes the task of creating or resetting the snap shot copies, without any need to modify the production volume extents as they exist in storage. The snapshot is reset by returning the entries in the pointer list to the lists of pointers to free tracks, and by clearing the bit map for the snapshot. If the snapshots for the files have all been created or updated in sequence, then the production volume extents can be restored with their snapshot copies. The restoration of each production volume extent can be performed by unlinking the entries from the list of pointers to the allocated storage locations in the snapshot copy, and for each entry, copying the pointed-to allocated track in the snapshot copy to the associated track of the production volume extent.

Detailed Description Text (69):
For debugging operations such as software simulation and testing, snapshot copies of files of interest can be made at various test points for inspection and comparison at a later time as problems are identified. In addition, the original contents of deleted or corrupted files can be restored from the snapshot copies. Shown in FIG. 17, for example, is a flowchart of a procedure programmed into a port adapter to give a host read access to a specified track of a snapshot of a production volume. In a first step 231, the port adapter inspects the bit for the track in the bitmap for the snapshot. If the bit for the track is set in the bit map, then execution continues from step 231 to step 232. In step 232, the port adapter accesses the index for the snapshot to find the snapshot volume number and the snapshot volume track number corresponding to the specified track of the snapshot of the production volume. Then in step 233, the track is read from the snapshot volume track, and the procedure of FIG. 17 is finished. If in step 231 the bit is not set for the track in the bit map, then execution branches to step 234. In step 234, the specified track is read from the production volume, and the procedure of FIG. 17 is finished.

CLAIMS:

1. In a data storage system including data storage, a method of maintaining in the data storage a snapshot copy of a production data set including a multiplicity of storage locations in the data storage, the production data set being accessible to a host processor for read/write access during maintenance of the snapshot copy, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, said method including: the data storage system responding to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the production data set includes at least a first production volume of tracks and a second production volume of tracks, the allocated storage location of the snapshot copy is in a snapshot volume of tracks, and the method further includes maintaining a first list of pointers to free tracks in the snapshot volume of tracks, maintaining a second list of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the first production volume, and maintaining a third list of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the second production volume, wherein the data storage system

allocates a track in the snapshot copy volume in response to the request from the host processor for a write operation to a track in a respective one of the production volumes by removing a pointer from the first list of pointers to free tracks in the snapshot volume and inserting the pointer on a respective one of the second and third lists of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the respective one of the production volumes.

2. A data storage system comprising data storage and at least one data processor responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the data processor being programmed to maintain in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the data processor is programmed to respond to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the production data set includes at least a first production volume of tracks and a second production volume of tracks, the allocated storage location of the snapshot copy is in a snapshot volume of tracks, and wherein the data processor is programmed to maintain a first list of pointers to free tracks in the snapshot volume of tracks, to maintain a second list of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the first production volume, and to maintain a third list of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the second production volume, and wherein the data processor is programmed to allocate a track in the snapshot copy volume in response to the request from the host processor for a write operation to a track in a respective one of the production volumes by removing a pointer from the first list of pointers to free tracks in the snapshot volume and inserting the pointer on a respective one of the second and third lists of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the respective one of the production volumes.

3. A program storage device containing a program for a data processor in a data storage system including data storage, the data processor being responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the program being executable by the data processor for maintaining in the data storage a snapshot copy of the production data set, the production data set includes at least a first production volume of tracks and a second production volume of tracks, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the program is executable by the data processor for responding to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, the allocated storage location of the snapshot copy being a track in a snapshot volume of tracks, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the program is executable by the data processor for maintaining a first list of pointers to free tracks in the snapshot volume of tracks, maintaining a second list of pointers to tracks in the snapshot

volume storing snapshot copies of tracks from the first production volume, and maintaining a third list of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the second production volume, and wherein the program is executable by the data processor for allocating a track in the snapshot copy volume in response to the request from the host processor for a write operation to a track in a respective one of the production volumes by removing a pointer from the first list of pointers to free tracks in the snapshot volume and inserting the pointer on a respective one of the second and third lists of pointers to tracks in the snapshot volume storing snapshot copies of tracks from the respective one of the production volumes.

4. In a data storage system including data storage, a method of maintaining in the data storage a snapshot copy of a production data set including a multiplicity of storage locations in the data storage, the production data set being accessible to a host processor for read/write access during maintenance of the snapshot copy, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, said method including: the data storage system responding to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created includes testing a bit for the storage location of the production data set, the bit being a bit of a bit map for the snapshot copy, and which includes migrating the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location that has not been modified in the production data set, copying to the secondary storage data from the storage location that has not been modified in the production data set, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

6. A data storage system comprising data storage and at least one data processor responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the data processor being programmed to maintain in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the data processor is programmed to respond to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created includes testing a bit for the storage location of the production data set, the bit being a bit of a bit map for the snapshot copy, and wherein the data processor is programmed to migrate the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location that has not been modified in the production data set, copying to secondary storage data from the storage location that has not been

modified in the production data set, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

8. A program storage device containing a program for a data processor in a data storage system including data storage, the data processor being responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the program being executable by the data processor for maintaining in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the program is executable by the data processor for responding to a request from the host processor for a write operation upon a storage location of the production data set by checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, copying data from the storage location of the production data set to an allocated storage location of the snapshot copy, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the program is executable by the data processor for checking whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created by testing a bit for the storage location of the production data set, the bit being a bit of a bit map for the snapshot copy, wherein the program is executable by the data processor for migrating the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location that has not been modified in the production data set, copying to secondary storage data from the storage location that has not been modified in the production data set, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

10. In a data storage system including data storage, a method of maintaining in the data storage a snapshot copy of a production data set, the production data set including a multiplicity of storage locations in the data storage, the production data set being accessible to a host processor for read/write access during maintenance of the snapshot copy, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, said method including: the data storage system allocating to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not said each storage location has been modified since the time when the snapshot copy is created, and the data storage system responding to a request from the host processor for a write operation upon a storage location of the production data set by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, which includes migrating the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location of the production data set that has not been modified

since the time when the snapshot copy was created, copying to the secondary storage data from the storage location of the production data set that has not been modified since the time when the snapshot copy was created, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

11. A data storage system comprising data storage and at least one data processor responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the data processor being programmed to maintain in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the data processor is programmed to allocate to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not said each storage location has been modified since the time when the snapshot copy is created, and wherein the data processor is programmed to respond to a request from the host processor for a write operation upon a storage location of the production data set by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the data processor is programmed to migrate the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location of the production data set that has not been modified since the time when the snapshot copy was created, copying to the secondary storage data from the storage location of the production data set that has not been modified since the time when the snapshot copy was created, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

13. A program storage device containing a program for a data processor in a data storage system including data storage, the data processor being responsive to requests from a host processor for read/write access to a production data set including multiple storage locations in the data storage, the program being executable by the data processor for maintaining in the data storage a snapshot copy of the production data set, the snapshot copy including data existing in the production data set at a time when the snapshot copy is created, wherein the program is executable by the data processor for allocating to the snapshot copy a bit map for the data set and a list of pointers, the bit map including a bit for each storage location of the production data set to indicate whether or not said each storage location has been modified since the time when the snapshot copy is created, and wherein the program is executable by the data processor for responding to a request from the host processor for a write operation upon a storage location of the production data set by checking the bit in the bit map for the storage location of the production data set to determine whether or not the storage location of the production data set has been modified since the time when the snapshot copy was created, and upon finding that the storage location of the production data set has not been modified since the time when the snapshot copy was created, allocating a storage location to the snapshot copy, copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, placing a pointer to the allocated storage location of the

snapshot copy on the list of pointers allocated to the snapshot copy, changing the bit in the bit map for the storage location of the production data set, and after copying data from the storage location of the production data set to the allocated storage location of the snapshot copy, performing the write operation upon the storage location of the production data set, wherein the program is executable by the data processor for migrating the snapshot copy from the data storage system to secondary storage by scanning the bit map, and for each bit in the bit map that indicates a storage location of the production data set that has not been modified since the time when the snapshot copy was created, copying to the secondary storage data from the storage location of the production data set that has not been modified since the time when the snapshot copy was created, and after scanning the bit map, transmitting data to the secondary storage from storage locations allocated to the snapshot copy.

Previous Doc Next Doc Go to Doc#

L5: Entry 1 of 1 File: USPT Apr 15, 2003

DOCUMENT-IDENTIFIER: US 6549992 B1
TITLE: Computer data storage backup with tape overflow control of disk caching of backup data stream

Brief Summary Text (18):
Since backup software need not place any constraints on the format of backup data other than it must be a stream of data that can be written to the tape device, the inventor has discovered that the performance of the storage system can be improved by the addition of certain facilities which may cause the tracks of a storage volume to become non-sequential as they are written to the tape device. In particular, it is desirable to continue host read-write access to a storage volume that is being backed up. If a host has read-write access to a storage volume, then the storage volume will be referred to as a "production volume." When a production volume in a primary storage subsystem is being backed up, it is desirable to receive the backup data from the primary storage subsystem as quickly as the primary storage subsystem delivers the backup data. Otherwise, when the host writes to the production volume, the maintenance of a snapshot copy of the backup data in the primary storage subsystem will increase the storage load and may also increase the processing load on the primary storage subsystem. Often, however, the tape storage device cannot write the backup data to tape as fast as the storage subsystem delivers the backup data. One solution to this problem is for the primary storage subsystem to write the backup data to intermediate disk storage in a secondary data storage subsystem and then write the backup data from the intermediate disk storage to tape storage. However, if the tracks of backup data from the snapshot copy of the production volume can be written to tape in a non-sequential fashion, then the required storage and data processing resources for the intermediate disk storage can be minimized by selectively bypassing-the intermediate disk storage whenever possible.

Drawing Description Text (7):
FIG. 5 is a block diagram of data structures which are included in a snapshot copy facility of the primary data storage subsystem of FIG. 1;

Drawing Description Text (8):
FIG. 6 is a schematic diagram of a preferred construction for a list of pointers to tracks in snapshot disks for the snapshot copy facility as shown in FIG. 5;

Drawing Description Text (9):
FIG. 7A is a flow chart of programming of the port adapters in the primary data storage subsystem of FIG. 3 for creating a snapshot of a production volume extent;

Drawing Description Text (10):
FIG. 7B is a flow chart of programming of the port adapters in the primary data storage subsystem of FIG. 3 for access to a production volume that is in a snapshot mode;

Drawing Description Text (11):
FIG. 8A is a flow chart of programming of the remote link adapters in the primary

data storage subsystem of FIG. 3 for a preferred implementation of the snapshot copy facility as introduced in FIG. 5;

Drawing Description Text (13):
FIG. 8C is a flow chart of programming of the remote link adapters in the primary data storage subsystem of FIG. 3 for an alternative implementation of the snapshot copy facility as introduced in FIG. 5;

Detailed Description Text (7):
In response to a backup command from the host 20, the primary data storage subsystem 21 accesses a primary directory 26 to find data of the physical storage unit specified by the backup command in order to initiate a process of copying the data from the primary storage 27 to the secondary storage 29 of the secondary data storage subsystem 22. Preferably, the primary directory 26 is constructed in such a way that the host can continue to access the primary storage 27 concurrently with the copying process. For example, in response to the backup command from the host 20, the primary data storage subsystem creates an "instant snapshot copy" of the specified physical storage unit, and this instant snapshot copy is protected from modification by the host 20 while the instant snapshot copy is being written to the secondary storage 29. There are a number of ways that such an instant snapshot copy can be created, depending on the way that the primary directory is organized.

Detailed Description Text (8):
One way of organizing the primary directory 26 is to associate a set of flags and mapping information with each physical storage unit, for example as described in Yanai et al., U.S. Pat. No. 5,206,939, issued Apr. 27, 1993, and incorporated herein by reference. In order to create an instant snapshot copy, a remote copy pending flag is associated with each physical storage unit. When the primary data storage subsystem 21 receives a backup command from the host 20, the primary data storage subsystem sets the remote copy pending flag, and thereafter the host can concurrently access the primary storage 27 while data is being copied from the physical storage unit to the secondary storage 29. However, before the primary data storage subsystem accesses the primary storage 27 to modify any data in response to a request from the host 20, the primary data storage subsystem first inspects the remote copy pending flag of the physical storage unit to be, modified, and if the remote copy pending flag is set, the primary data storage subsystem must copy the data of the physical storage unit from the primary storage 27 to the secondary storage 29 and reset the remote copy flag, before modifying the data in the primary data storage subsystem. Unless there is such a request from the host for modification of data marked as "remote copy pending," the copying of data from the primary storage 27 to the secondary storage 29 is performed as a background process relative to host requests.

Detailed Description Text (9):
Another way of organizing the primary directory 26 is to maintain lists of pointers to primary storage locations of old and new versions of physical storage units. Whenever data in the primary storage is modified, the data is read from the primary storage locations containing the most current version, modified, and written to a different set of primary storage locations. This is known in the art as a "log structured file" approach. See, for example, Douglis et al. "Log Structured File Systems," COMPCON 89 Proceedings, Feb. 27-Mar. 3, 1989, IEEE Computer Society, p. 124-129, incorporated herein by reference, and Rosenblum et al., "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, Vol. 1, February 1992, p. 26-52, incorporated herein by reference. In order to create an instant snapshot copy, the primary directory 26 includes a respective remote copy pending flag associated with the pointer to each version of each physical storage unit. In response to a backup command from the host 20, the primary data storage subsystem sets the remote copy flag. Thereafter, the primary data storage subsystem can modify the data of the physical storage unit in the primary storage in the usual fashion concurrently with the copying of a snapshotted

version of the data to the secondary storage, because the new version and the snapshotted version of the physical storage unit are stored in a different set of primary storage locations. Instead of being used to temporarily delay any modification access to a physical storage unit, the remote copy flag is used to indicate whether or not the set of primary storage locations associated with an old version of a physical storage unit can be de-allocated after a modification access.

Detailed Description Text (11):
In the preferred implementation of the data processing system of FIG. 1, the primary data storage subsystem 21 includes a snapshot copy facility 69. The snapshot copy facility 69 includes a stored program that is executed by data processors in the primary data storage subsystem as described below with reference to FIGS. 5 to 8. This stored program is a component of what is known as microcode for the primary data storage subsystem. The microcode can be down-loaded into program memory of the primary data storage subsystem from a conventional program storage device such as a floppy disk.

Detailed Description Text (12):
Regardless of how the primary directory 26 is organized and how the instant snapshot process is performed, it is possible for the secondary storage 29 to contain more than one version of backup data for the same physical storage unit. In order to distinguish between different versions of backup data for the same physical storage unit, the primary data storage subsystem 21 appends an identification tag to the backup data transmitted from the primary data storage subsystem to the secondary data storage subsystem 22. The tag, for example, is supplied by the host 20 in the backup command transmitted by the host to the primary data storage subsystem 21. The tag could also include a date-time stamp generated by the primary data storage subsystem. In the secondary data storage subsystem 22, the tag associated with each version of backup data is stored in a secondary directory 28, which further includes a record of a set of locations of the secondary storage 29 in which the version of backup data is stored.

Detailed Description Text (16):
With reference to FIG. 2, a data network 30 such as a Fibre Channel loop links a multiplicity of hosts 31, 32, 33 to a number of primary data storage subsystems 41, 42. The hosts 31, 32, 33, for example, are workstations of respective users 34, 35, 36. The user 35 is a system manager responsible for configuring the data storage subsystems 41, 42 and ensuring that the data storage and backup needs of the users are satisfied. Each of the hosts has a copy of backup software 37, 38, 39 similar to the backup software 24 described above with reference to FIG. 1. The primary data storage subsystems 41,42 each have a respective primary directory 44, 46, respective primary storage 45, 47, and respective snapshot copy facilities 67, 68. The primary data storage subsystems 41, 42 are each similar to the primary data storage subsystem 21 of FIG. 1. The primary data storage subsystems 41, 42 share a secondary data storage subsystem 43. The secondary data storage subsystem 43 has a secondary directory 48, secondary storage 49, and a disk caching facility 69. The secondary data storage subsystem 43 is similar to the secondary data storage subsystem 22 of FIG. 1, but it further includes independent, dedicated data links 93 and 94 to each of the primary data storage subsystems 41 and 42 and a data link 92 to the data network 30. The dedicated links 93, 94 are used for transferring backup data between the respective primary data storage subsystems 41, 42 and the secondary data storage subsystem 43. The data link 92 permits the system manager 35 to access the secondary data storage subsystem 43 for data storage management and diagnostic purposes.

Detailed Description Text (31):
II. Snapshot Copy Facility

Detailed Description Text (32):

As described above with reference to FIGS. 1 to 4, a primary data storage subsystem and a secondary data storage subsystem have been constructed to rapidly respond to a backup request. The primary data storage subsystem responds by performing a snapshot copy, and transferring backup data from the snapshot copy to the secondary storage subsystem.

Detailed Description Text (33):
With reference to FIG. 5, there are shown a number of data structures that are located in the primary data storage subsystem and are used by the snapshot copy facility (69 in FIG. 1) of the primary data storage subsystem. In the example of FIG. 5, snapshot copies have been made of two production volumes 101 and 102. The snapshot copies are shown as they would exist some time after the primary storage subsystem has received a first backup command for backing up an "extent" of the production volume 101, and some time after the primary data storage subsystem has received a second backup command for backing up an extent of the production volume 102. An "extent" of a production volume is a set of contiguous tracks of the production volume, as specified, for example, by a beginning track number and an ending track number. Since receipt of the first backup command, a host has modified tracks A and B of the production volume 101, and since receipt of the second backup command, a host has modified tracks G and H of the production volume 102. Before the first modification of each track in the snapshotted production volume 101 or 102, however, the primary data storage subsystem copies the original contents of the production volume (i.e., the contents existing at the time of the snapshot) to a track in a snapshot volume 103, 104. For example, before the primary data storage subsystem modifies track A of the production volume 101, the original contents of track A are copied to track 0 of the snapshot volume 103. In a similar fashion, track 1 of the snapshot volume 103 contains the original contents of track B of the production volume 101, track 2 of the snapshot volume 103 contains the original contents of track G of the production volume 102, and track 3 of the snapshot volume 103 contains the original contents of track H of the production volume 102.

Detailed Description Text (35):
The data structures in FIG. 5 also include, for each snapshotted production volume extent, a list of pointers 106, 108 to tracks in the snapshot volumes that contain original data of the snapshot. The data structures also include, for each snapshot volume, a list of pointers 109, 110 to free tracks. When a track is copied from an extent of a production volume to a snapshot volume, a pointer to the track is taken from the list of pointers to free tracks in the snapshot volume and added to the list of pointers for the extent.

Detailed Description Text (36):
When a snapshot copy of a production volume is first created, the directory to the production volume is locked to host access while the bit map and list of pointers to snapshot tracks are allocated and initialized for the snapshot copy, and then the directory is unlocked.

Detailed Description Text (37):
The data structures for the snapshot copy facility as shown in FIG. 5 permit the production volumes to be configured and indexed in the usual fashion. The tracks of the snapshot copy for an extent can be obtained by scanning the bit map for the extent and accessing the production volume for a track having its respective bit not set in the bit map, and otherwise accessing the track in the snapshot volume using a pointer from the list of pointers for the extent if the respective bit for the track is set. The data structures used by the snapshot copy facility include information associating the tracks for the extent with their respective pointers in the list.

Detailed Description Text (38):
Although not necessary for making backup copies, the data structures associated with the snapshot copy facility may include an index to the snapshot tracks for

each snapshot copy of a production volume. Such an index, for example, would be desirable if the snapshot copy facility were used for providing specified snapshot data to a distributed transaction processing system. It could also be useful if it were desired to transmit snapshot copy data from the primary storage subsystem in sequential track order, or to de-allocate specified snapshot tracks. As shown in FIG. 5, for example, an index 111 is provided to locate, in the list of pointers 106, the pointer to any snapshot track containing original data from a specified track in the production volume 101. Also, an index 112 is provided to locate, in the list of pointers 108, the pointer to any snapshot track containing original data from a specified track in the production volume 102. In other words, each index 111, 112 functions as a kind of translation table, to translate a production volume track number to a corresponding snapshot volume number and snapshot volume track number.

Detailed Description Text (39):
Although the indices 111 and 112 can be constructed in various ways, in many applications only a small fraction of the production volume tracks will have corresponding snapshot volume tracks. In this situation, a conventional hash table index is an efficient implementation for the indices 111, 112. For the index 111, for example, such a conventional hash table index includes, for each possible hashing of the production volume track number, a list of pointers to corresponding entries in the list 106. Therefore, given a specified production volume track number, any corresponding snapshot track is found by hashing the production volume track number (i.e., obtaining the remainder of the division of the production volume track number by a predetermined number of possible hashings), and then searching the corresponding entries in the list 106 to locate an entry for the specified production volume track number.

Detailed Description Text (40):
As shown in FIG. 6, the list of pointers 106 includes heading information that specifies the list's production volume extent. This heading information includes a logical device number (DEVICE_NUM), a first track number for the extent (FIRST_TRACK), and a last track number for the extent (LAST_TRACK). Each entry in the list includes a snapshot volume number (X), a production track number (A, B), and a snapshot track number (0, 1). In this example, the list is maintained as a singly-linked list, so that the header information includes a pointer to a first entry in the list (or has a value of zero if the list is empty), and each entry in the list includes a pointer to a next entry in the list (and this pointer has a value of zero for the last entry in the list).

Detailed Description Text (41):
When the snapshot copy facility is used to facilitate backup operations, it is desirable to use a singly-linked list of pointers 106 instead of simply a list in the form of an array in order to conserve memory for the list because the length of the list is highly variable and it is possible that multiple snapshot copies may be in existence simultaneously. For example, the tape library unit has multiple read/write stations (81, 82, 83, 84 in FIG. 4) which may be writing backup data simultaneously to different respective tape cassettes, and each production volume extent is written to a respective tape cassette.

Detailed Description Text (42):
Referring to FIG. 7A, there is shown a flow chart of a software procedure programmed in the port adapters of a primary data storage subsystem for creating a snapshot copy of a production volume. The procedure of FIG. 7A is invoked, for example, when the port adapter receives a command from a host requesting backup of a specified production volume extent. In the first step 120 of FIG. 7A, the port adapter allocates the snapshot data structures, including a bit map and a list of pointers to snapshot tracks for the extent. An index to the pointers may also be allocated. Then in step 121, the port adapter inserts, into the volume directory entry for the production volume, a pointer to the snapshot data structures. This

pointer, for example, is zero in the volume directory entry for any volume which does not have a snapshot copy. The volume directory entry could also have a field specifying a minimum track number for the production volume extent to be snapshotted, and a maximum track number for the production volume extent to be snapshotted. After step 121, the snapshot creation procedure is finished.

Detailed Description Text (43):
Referring to FIG. 7B, there is shown a flowchart of a software procedure programmed into the port adapters for accessing a production volume in a snapshot mode. This procedure is invoked whenever a host processor requests a write operation to a production volume for which a snapshot could have been created. In the first step 122 the port adapter checks the volume director entry to determine whether a snapshot currently exists for the production volume, and if so whether the write operation is upon a track within the production volume extent of the snapshot. If the access to the production volume is not a write to a track within the production volume extent of the snapshot, then execution branches to step 123 to access the track in the production volume, and then the procedure of FIG. 7B is finished. Otherwise, if the access to the production volume is a write to a track within the production volume extent of the snapshot, then execution continues from step 122 to step 124. In step 124, the port adapter inspects the bit for the track in the bit map for the extent. If the bit is set, then execution branches to step 123 to access the track in the production volume extent, and then the procedure of FIG. 7B is finished. Otherwise, if the bit for the track is not set in the bit map, then execution continues from step 124 to step 125. In step 125 the port adapter obtains a pointer to a free track in one of the snapshot volumes. For example, the port adapter first checks whether the list of pointers 109 for a first volume 103 is empty, and if so, it then checks the list of pointers for other snapshot volumes such as the volume 104 in FIG. 5.

Detailed Description Text (44):
Once the port adapter finds a non-empty list of pointers to free tracks of a snapshot volume, it removes a pointer from the list. In step 126 the port adapter copies the track to be modified from the production volume to the snapshot volume track specified by the pointer that was taken from the list of pointers to free tracks in the snapshot volume. In step 127 the port adapter inserts the pointer into the list of snapshot track pointers for the extent, and also inserts into this list entry an identifier for the snapshot volume and an identifier for the track in the production volume extent. Then in step 128 the port adapter sets the bit in the bit map to indicate that the track is being modified. Finally, in step 129 the port adapter writes new data to the track in the production volume. After step 129, the procedure of FIG. 7B is finished.

Detailed Description Text (45):
With reference to FIG. 8A, there is shown a flow chart of a software procedure programmed in a remote link adapter of a primary storage subsystem for implementation of the snapshot copy facility. This procedure is invoked in response to a message from a port adapter when the port adapter receives a backup command from a host. In a first step 131, the remote link adapter sets a track copy pointer to point to the first track in the production volume extent. Then in step 132 the remote link adapter checks whether the bit for the track is set in the bit map for the production volume extent. If not, then in step 133 the remote link adapter copies the track from the production volume to secondary storage. Execution continues from step 133 to step 134. Execution also branches from step 132 to step 134 when the remote link adapter finds in step 132 that the bit for the track has been set in the bit map. In step 134, the remote link adapter checks whether the track copy pointer is pointing to the end of the production volume extent. If not, execution continues from step 134 to step 135. In step 135 the track copy pointer is incremented to point to the next track in the extent, and execution loops back to step 132.

Detailed Description Text (46):
If in step 134 the remote link adapter finds that the track copy pointer is pointing to the end of the production volume extent, then execution branches to step 136 In step 136, the remote link adapter deallocates the bit map. Then in step 137, for each entry in the list of pointers to snapshot tracks, the remote link adapter copies the track from the snapshot volume to secondary storage, and removes the entry from the list. Finally, in step 138, the remote link adapter deallocates the list of pointers to snapshot tracks for the extent, and the backup operation is finished.

Detailed Description Text (47):
The remote link adapter routine of FIG. 8A may copy the tracks of snapshot data in either a synchronous fashion or an asynchronous fashion from the primary data storage subsystem to the secondary storage subsystem. If the copying is done in a synchronous fashion, the procedure in FIG. 8A would not proceed from step 133 to step 134 until the remote link adapter would receive confirmation from the secondary storage that the track has in fact been written to the secondary storage. Preferably, however, the copying is done in an asynchronous fashion, in which the track is transmitted from the production volume to secondary storage in step 133 and execution then continues to step 134 without waiting for confirmation that the that the track has been written to the secondary storage. Once the remote link adapter receives confirmation that a track has been written from the production volume to secondary storage, the bit for the track in the bit map can be set to avoid any delay if and when the host makes any write access to the production volume track before the entire production volume extent has been copied to secondary storage. Also, if a backup copy is made in an asynchronous fashion, the bit map is not deallocated in step 136 until after receipt of confirmation that all of the tracks sent from the production volume to secondary storage in step 133 have actually been written to secondary storage.

Detailed Description Text (48):
To facilitate backup copying to secondary storage in an asynchronous fashion, the remote link adapter can be programmed as shown in FIG. 8B to respond to a command from the secondary data storage system indicating that a range of specified tracks have been in fact copied to secondary storage and therefore the specified tracks can be deallocated from the snapshot copy. The range of specified tracks, for example, is a small subset of the tracks in the production volume extent being backed up. In this case, the command from the secondary storage is a request to "partially free" the snapshot copy. The deallocation of the specified range of tracks will prevent copying of production volume tracks in the specified range to snapshot tracks if a host subsequently writes to the tracks for the first time after the creation of the snapshot copy, and will free any snapshot volume tracks that have been allocated to production volume tracks within the specified range.

Detailed Description Text (49):
In a first step 221 of FIG. 8B, the remote link adapter sets a deallocation track pointer to the first track in the specified range of tracks. Then in step 222, execution branches depending on the state of the bit for the track in the bit map. If the bit for the track is set in the bit map, then execution branches from step 222 to step 223. In step 223, any snapshot volume track allocated for the production volume track is deallocated by removing the snapshot track pointer from the list for the snapshot volume and returning the snapshot track pointer to the free list for the snapshot volume. If in step 222 the bit for the track is not set in the bit map, then execution continues from step 222 to step 224. In step 224, the bit for the track is set in the bit map. Therefore, if a host subsequently writes to the track, a copy of the original contents of the track will not be copied to a snapshot volume track. After step 223 or 224, execution continues to step 225. In step 225, the deallocation track pointer is compared to the end of the specified range. If the deallocation track pointer is at the end of the specified range, then the procedure is finished. If not, execution continues to step 226. In

step 226, the deallocation track pointer is incremented, and execution loops back to step 222.

Detailed Description Text (51):
Alternatively, the tracks can be copied from a snapshot to secondary storage in a sequential fashion. In an example of such an alternative procedure, as shown in the flow chart of FIG. 8C, the link adapter is programmed to respond to a backup request by sequentially incrementing a track copy pointer from the beginning track in a production volume extent to an ending track in the production volume extent. In a first step 241, the link adapter sets the track copy pointer to point to the first track in the production volume extent. Then, in step 242, for the track pointed to by the track copy pointer, the link adapter indexes the bit map for the snapshot. If the bit for the track is set in the bitmap for the snapshot, then execution continues from step 242 to step 243. In step 243, the link adapter accesses the index to the snapshot tracks to translate the track number specified by the track copy pointer to a snapshot volume number and a snapshot track number. Then in step 244, the link adapter reads the snapshot track from the snapshot volume and transmits this track of backup data from the snapshot track to the secondary storage subsystem.

Detailed Description Text (52):
If the bit for the track is not set in the bit map for the snapshot, then execution branches from step 242 to step 245. In step 245, the link adapter reads the track specified by the track copy pointer from the production volume and transmits this track of backup data to the secondary storage subsystem. After step 244 or 245, execution continues to step 246. In step 246, if the track copy pointer has not reached the end of the production volume extent, then execution branches to step 247. In step 247, the track copy pointer is incremented, and execution loops back to step 242. Once the track copy pointer reaches the end of the production volume extent in step 246, the procedure of FIG. 8C is finished.

Detailed Description Text (53):
With reference to FIG. 9, there is shown a format of a data record on the backup tape. In accordance with a conventional tape record, the record shown in FIG. 9 includes, after an inter-record gap 141, a synchronization code 142, a record number 143, record data (fields 144, 145, and 146), and finally an error correction code 147 preceding another inter-record gap 148. Since the records on the backup tape are not necessarily sequential by track number, it is desirable for each record to include a track number 145 in the record data. Also, it is assumed that each tape cassette will include data from only one production volume extent. It is possible, however, that a tape cassette could be used, at various times, to store more than one version of data from the same production volume extent. Therefore, it is also desirable for the data of the record on tape to include a version identifier 144, such as a date/time stamp when the snapshot copy was made. As shown in FIG. 9, the data portion of the record on the backup tape includes the version identifier 144 followed by the track number 145 and track data 146.

Previous Doc Next Doc Go to Doc#

L5	L4 and refresh$	8	L5
L4	L3 and production	25	L4
L3	L2 and (read near write)	164	L3
L2	L1 and (snapshot near copies)	211	L2
L1	file near system	36476	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs
Generate OACS

Search Results - Record(s) 1 through 8 of 8 returned.

☐ 1. Document ID: US 20050066095 A1

```
L5: Entry 1 of 8                    File: PGPB                  Mar 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050066095
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050066095 A1

TITLE: Multi-threaded write interface and methods for increasing the single file
read and write throughput of a file server

PUBLICATION-DATE: March 24, 2005

INVENTOR-INFORMATION:
NAME                 CITY            STATE     COUNTRY     RULE-47
Mullick, Sachin      Ashland         MA        US
Zheng, Jiannan       Ashland         MA        US
Jiang, Xiaoye        Shrewsbury      MA        US
Faibish, Sorin       Newton          MA        US
Bixby, Peter         Westborough     MA        US

US-CL-CURRENT: 710/200
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☑ 2. Document ID: US 20050065986 A1

```
L5: Entry 2 of 8                    File: PGPB                  Mar 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050065986
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050065986 A1

TITLE: Maintenance of a file version set including read-only and read-write
snapshot copies of a production file

PUBLICATION-DATE: March 24, 2005

INVENTOR-INFORMATION:
NAME                 CITY            STATE     COUNTRY     RULE-47
Bixby, Peter         Westborough     MA        US
Mullick, Sachin      Ashland         MA        US
```

PGPUB-DOCUMENT-NUMBER: 20040030727
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040030727 A1

TITLE: Organization of multiple snapshot copies in a data storage system

PUBLICATION-DATE: February 12, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Armangau, Philippe	Acton	MA	US	
Bergant, Milena	North Grafton	MA	US	
Wang, Hongmei	Shrewsbury	MA	US	
Potnis, Ajay	Sauth Grafton	MA	US	
Angelone, Raymond A.	Norfolk	MA	US	

US-CL-CURRENT: 707/200

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☑ 8. Document ID: US 6792518 B2

L5: Entry 8 of 8 File: USPT Sep 14, 2004

US-PAT-NO: 6792518
DOCUMENT-IDENTIFIER: US 6792518 B2

TITLE: Data storage system having mata bit maps for indicating whether data blocks are invalid in snapshot copies

DATE-ISSUED: September 14, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Armangau; Philippe	Acton	MA		
Tummala; Himabindu	South Grafton	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 10/ 213241 [PALM]
DATE FILED: August 6, 2002

INT-CL: [07] G06 F 12/00

US-CL-ISSUED: 711/162; 711/112, 711/156, 707/203, 707/204, 714/5
US-CL-CURRENT: 711/162; 707/203, 707/204, 711/112, 711/156, 714/5

FIELD-OF-SEARCH: 707/203, 707/204, 711/112, 711/156, 711/161, 711/162, 714/5

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4608688	August 1986	Hansen et al.	371/11
4686620	August 1987	Ng	
4755928	July 1988	Johnson et al.	
5060185	October 1991	Naito et al.	
5206939	April 1993	Yanai et al.	
5381539	January 1995	Yanai et al.	
5535381	July 1996	Kopper	
5596706	January 1997	Shimazaki et al.	395/182.04
5673382	September 1997	Cannon et al.	
5737747	April 1998	Vishlitzky et al.	711/118
5742792	April 1998	Yanai et al.	
5819292	October 1998	Hitz et al.	707/203
5829046	October 1998	Tzelnic et al.	711/162
5829047	October 1998	Jacks et al.	711/162
5835954	November 1998	Duyanovich et al.	711/162
5915264	June 1999	White et al.	711/168
5923878	July 1999	Marsland	395/704
5974563	October 1999	Beeler, Jr.	714/5
6061770	May 2000	Franklin	711/162
6076148	June 2000	Kedem	711/162
6078929	June 2000	Rao	707/200
6269431	July 2001	Dunham	711/162
6279011	August 2001	Muhlestein	707/204
6434681	August 2002	Armangau	711/162
6618794	September 2003	Sicola et al.	711/154
2003/0079102	April 2003	Lubbers et al.	711/202
2003/0158873	August 2003	Sawdon et al.	707/204

OTHER PUBLICATIONS

Mendel Rosenblum and John K. Ousterhout, "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, vol. 10, No. 1, Feb. 1992, pp. 26-52.
Fred Douglis and John Ousterhout, "Log-Structured File Systems," in Spring COMPCON89, Feb. 27-Mar. 31, 1989, Thirty-Fourth IEEE Computer Society International Conference, San Francisco, CA, pp. 124-129.
David A. Patterson, Peter Chen, Garth Gibson, and Randy H. Katz, "Introduction to Redundant Arrays of Inexpensive Disks (RAID)," in Spring COMPCON89, Feb. 27-Mar. 31, 1989, Thirty-Fourth IEEE Computer Society International Conference, San Francisco, CA, pp. 112-117.
D.L. Burkes and R.K. Treiber, "Design Approaches for Real-Time Transaction Processing Remote Site Recovery," in Spring COMPCON90, Feb. 26-Mar. 2, 1990, Thirty-Fifth IEEE Computer Society International Conference, San Francisco, CA, pp. 568-572.
"VERITAS NetBackup and Storage Migrator" http://www.sun.com/stora . . . /netbackup.html; $sessionid$QEOQTDQAAC2QHAMTA1FU5Y, published at least as early as Oct. 28, 2000, 5 pages.
R. Stager and D. Hitz, Internet Draft, filename "draft-stager-iquard-netapp-backup-

0.5.txt" Network Data Management Protocol (NDMP), last update Oct. 12, 1999, pp. 1-73.
"Network Data Management Protocol (NDMP)," http://www.ndmp.org/info/; NDMP White Paper, http://www.ndmp.org/info/technology/wp.html; "Protocol Specification Summary, Document Version: 1.7.2S," http://www.ndmp.org/info/spec_summary.html; "Legato Systems Embraces the NDMP Standard: Legato Networker Software to be NDMP Compliant in Q3," http://www-ftp.legata.com/News/Press/PR209.html; published at least as early as Oct. 11, 1999, 17 pages.
"RFC 1094--NFS: Network File System Protocol Specification," Network Working Group, Request for Comments: 1094, Sun Microsystems, Inc., Mar. 1989, pp. 1-27, http://rfc.sunsite.dk/rfc/rfc1094.html.
Uresh Vahalia, Unix Internals--The New Frontiers, Prentice-Hall Inc., New Jersey, 1996, Chapter 9, File System Implementations, pp. 261-289.
Brian W. Kerningham and Rob Pike, The UNIX Programming Environment, Prentice-Hallz Inc., New Jersey, 1984, Chapter 2, The File System, pp. 41-70.
Koop, P., "Replication at Work. (four companies use Oracle and Sybase replication servers to solve business problems)," DBMS, vol. 8, No. 3, p. 54(4), Mar. 1995.
Remote Mirroring Technical White Paper, Copyright 1994-2002 Sun Microsystems, published at least as early as May 17, 2002 at sun.com, 25 pages.
EMC TimeFinder Product Description Guide, EMC Corporation, Hopkinton, MA, 1998, pp. 1-31.
Leveraging SnapView/IP in Oracle8i Environments with the CLARiiON IP4700 File Server, Engineering White Paper, EMC Corporation, Hopkinton, MA, Feb. 13, 2002, pp. 1-16.
Using EMC CLARiiON FC4700 and SnapView with Oracle 8i, Engineering White Paper, EMC Corporation, Hopkinton, MA, Mar. 4, 2002, pp. 1-22.
Disaster Recovery Guidelines for using HP SureStore E XP256, Continuous Access XP with Oracle Databases Rev 1.03, Hewlett-Packard Company, Palo Alto, CA, May 2000, pp. 1-28.
Enterprise Volume Manager and Oracle8 Best Practices, Compaq White Paper, Compaq Computer Corporation, Dec. 1999, pp. 1-11.
VERITAS Database Edition for Oracle, Guidelines for Using Storage Checkpoint and Storage Rollback with Oracle Databases, Veritas Software Corporation, Mountain View, CA, Aug. 2001, pp. 1-16.
VERITAS Volume Replication and Oracle Databases, A Solutions White Paper, Veritas Software Corporation, Mountain View, CA, May 29, 2000, pp. 1-31.
Nabil Osorio and Bill Lee, Guidelines for Using Snapshot Storage Systems for Oracle Databases, Oracle Corporation, Oct. 2001, pp. 12.

ART-UNIT: 2187

PRIMARY-EXAMINER: Moazzami; Nasser

ASSISTANT-EXAMINER: Chace; Christian P.

ATTY-AGENT-FIRM: Novak Druce LLP Auchterlonie; Richard C.

ABSTRACT:

In a data storage system providing access to a production dataset and snapshot copies of the production dataset, a production meta bit map identifies blocks of storage that are invalid in the production dataset. If a block in the production dataset is invalid when a snapshot copy is being made, then there is no need to copy the block to storage for the snapshot before writing to the block. Moreover, if a block in the production dataset supporting a snapshot copy is dynamically invalidated, it may be kept in the production dataset until it is written to. For this purpose, a respective snapshot copy of the meta bit map is made and kept with each of the snapshot datasets, and the snapshot copies of the meta bit map are merged in order to indicate the blocks that are invalid for all of the snapshots.

```
46 Claims, 40 Drawing figures
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. D

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Term	Documents
REFRESH$	0
REFRESH	44153
REFRESHA	1
REFRESHAB	1
REFRESHABILITY	10
REFRESHABLE	412
REFRESHABLY	2
REFRESHABORTED	1
REFRESHABORTEDEXCEPTION	6
REFRESHACCESS	1
REFRESHACTION	1
(L4 AND REFRESH$).PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD.	8

There are more results than shown above. Click here to view the entire set.

Display Format: - Change Format

Previous Page Next Page Go to Doc#

Zheng, Jiannan	Ashland	MA	US
Jiang, Xiaoye	Shrewsbury	MA	US
Faibish, Sorin	Newton	MA	US

US-CL-CURRENT: 707/204

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. De

---

☐ 3. Document ID: US 20050065985 A1

L5: Entry 3 of 8 File: PGPB Mar 24, 2005

PGPUB-DOCUMENT-NUMBER: 20050065985
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20050065985 A1

TITLE: Organization of read-write snapshot copies in a data storage system

PUBLICATION-DATE: March 24, 2005

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Tummala, Himabindu	South Grafton	MA	US	
Armangau, Philippe	Acton	MA	US	

US-CL-CURRENT: 707/201

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. De

---

☑ 4. Document ID: US 20040107222 A1

L5: Entry 4 of 8 File: PGPB Jun 3, 2004

PGPUB-DOCUMENT-NUMBER: 20040107222
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040107222 A1

TITLE: Client-server protocol for directory access of snapshot file systems in a storage system

PUBLICATION-DATE: June 3, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Venkatesh, Dinesh	Westford	MA	US	
Jiang, Xiaoye	Shrewsbury	MA	US	
Zheng, Jiannan	Ashland	MA	US	
Vahalia, Uresh	Newton	MA	US	

US-CL-CURRENT: 707/200

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☑ 5. Document ID: US 20040030951 A1

```
L5: Entry 5 of 8                    File: PGPB                  Feb 12, 2004
```

PGPUB-DOCUMENT-NUMBER: 20040030951
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040030951 A1

TITLE: Instantaneous restoration of a production copy from a snapshot copy in a data storage system

PUBLICATION-DATE: February 12, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Armangau, Philippe	Acton	MA	US	

US-CL-CURRENT: 714/6

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 6. Document ID: US 20040030846 A1

```
L5: Entry 6 of 8                    File: PGPB                  Feb 12, 2004
```

PGPUB-DOCUMENT-NUMBER: 20040030846
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040030846 A1

TITLE: Data storage system having meta bit maps for indicating whether data blocks are invalid in snapshot copies

PUBLICATION-DATE: February 12, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Armangau, Philippe	Acton	MA	US	
Tummala, Himabindu	South Grafton	MA	US	

US-CL-CURRENT: 711/154; 711/159

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☑ 7. Document ID: US 20040030727 A1

```
L5: Entry 7 of 8                    File: PGPB                  Feb 12, 2004
```